VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Multi-senzorový digitální teploměr
Digital Thermometer with Multiple Sensors

2011 Michal Chromec

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Michal Chromec**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Multi-senzorový digitální teploměr
Digital Thermometer with Multiple Sensors

Zásady pro vypracování:

Cílem práce je konstrukce multisenzorového digitálního teploměru, který bude umožňovat měření teploty v rozmezí -40°C až +125°C na více místech v prostoru. Dále bude umět naměřené hodnoty zobrazit na připojené zobrazovací jednotce a nad definovanými skupinami teplotních čidel počítat průměrnou teplotu. Zařízeni bude rovněž schopno naměřené hodnoty poskytovat pomocí sériového rozhraní.

1) Navrhněte konstrukci digitálního teploměru s více teplotními čidly a zobrazovací jednotkou.
2) Navržené řešení zkonstruujte.
3) Implementujte vhodný firmware, který by kromě měření teploty a jejího případného přenosu po sériovém rozhraní, umožňoval průměrové výpočty nad definovanými skupinami získaných hodnot z teplotních senzorů s následným zobrazením.
4) Vzniklé řešení důkladně otestujte.

Seznam doporučené odborné literatury:

MATOUŠEK, David. Práce s mikrokontroléry Atmel AVR. Praha : BEN - technická literatura, 2006. 376 s. ISBN 80-7300-209-4.
Maxim Integrated Products [online]. USA : Maxim Integrated Products, 2008 [cit. 2010-11-07]. DS18B20 Programmable Resolution 1-Wire Digital Thermometer. Dostupné z WWW: <http://pdfserv.maxim-ic.com/en/ds/DS18B20.pdf>.
MATOUŠEK, David. Práce s inteligentními displeji LCD . Praha : BEN - technická literatura, 2006. 224 s. ISBN 80-7300-121-7.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Martin Milata**

Datum zadání: 19.11.2010

Datum odevzdání: 06.05.2011

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení Studenta

„Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.“

V Ostravě 5. května 2011

Abstrakt

Tato bakalářská práce se zabývá vytvořením multi-senzorového digitálního teploměru. Teploměr obsahuje pět sběrnic 1-Wire, na které je možné libovolně připojit čidla Dallas DS18B20. Po připojení napájení nebo při resetování zařízení si automaticky projde všechny sběrnice a zjisti připojená čidla. Dále teploměr zobrazuje teploty vybraných čidel na LCD displeji. Další funkcí teploměru je práce se skupinami teplotních čidel, nad kterými počítá průměrnou teplotu. Teploměr podporuje připojení k počítači přes USB a poskytnutí naměřených i průměrných hodnot teploty. Zobrazení teplot na počítači umožňuje PC aplikace, pomocí které lze také teploměr nastavit a definovat průměrové skupiny. Hlavním prvkem teploměru je mikrokontroler Atmel ATmega8L, který řídí všechny ostatní obvody a komunikuje s teplotními čidly. Komunikaci mezi mikrokontrolerem a USB zajišťuje převodník FTDI Chip FT232RL.

Abstract

This bachelor work deal with creation thermometer with multiple sensors. Thermometer contains five 1-Wire buses, at what we can connect sensors Dallas DS18b20. When is connected power supply or device is reset, thermometer go over all buses and find out connected sensors. Thermometer displays temperature of selected sensors on LCD. Next function of thermometer is working with groups of temperature sensors, which can compute average temperature. Thermometer supports connecting to PC by USB and provides measured temperature and average temperature. Display temperatures on PC allow PC application, which can configure thermometer and define groups of sensors. The main part of thermometer is microcontroller Atmel ATmega8L, which controls other electronic device and communicates with temperature sensors. The communication between microcontroller and USB is provided by converter FTDI Chip FT232RL.

Klíčová slova

Digitální teploměr, teplotní čidlo, mikrokontroler, Atmel, USB, DS18B20

Keywords

Digital thermometer, thermometer sensor, microcontroller, Atmel, USB, DS18B20

Seznam použitých symbolů a zkratek

AVR – označení jádra mikrokontrolerů Atmel

C – programovací jazyk

C# - programovací jazyk

DPS – deska plošného spoje

LCD – displej z tekutých krystalů

PC – osobní počítač

RISC – redukovaná instrukční sada procesoru

ROM – paměť určená jen pro čtení

USART – univerzální synchronní asynchronní přijímač vysílač

USB – univerzální sériová sběrnice

Obsah

1 Úvod

Cílem této bakalářské práce bylo vtvořit multi-senzorový digitální teploměr. V práci je postupně uvedeno, jak byl teploměr vytvořen. První část obsahuje informace o jednotlivých částech, ze kterých byl teploměr sestaven. Zde jsou uvedeny informace o použitých teplotních čidlech, řídícím mikrokontroleru, obvodu pro sériovou komunikaci a zobrazovací jednotce. V další části je popis implementace firmware pro mikrokontroler a implementace aplikace pro PC, která bude umožňovat zobrazení naměřených teplot, a také bude sloužit k nastavení teploměru. Nastavení bude řešit, ze kterých čidel se bude zobrazovat teplota na zobrazovací jednotce, dále se budou definovat čidla, z jejichž naměřených teplot se bude počítat průměr.

2 Popis jednotlivých částí teploměru

2.1 Teplotní čidlo DALLAS DS18B20

Dallas DS18B20 je digitální teplotní čidlo s rozlišením 9 až 12 bitů. Komunikace s čidlem probíhá po 1-Wire sběrnici.

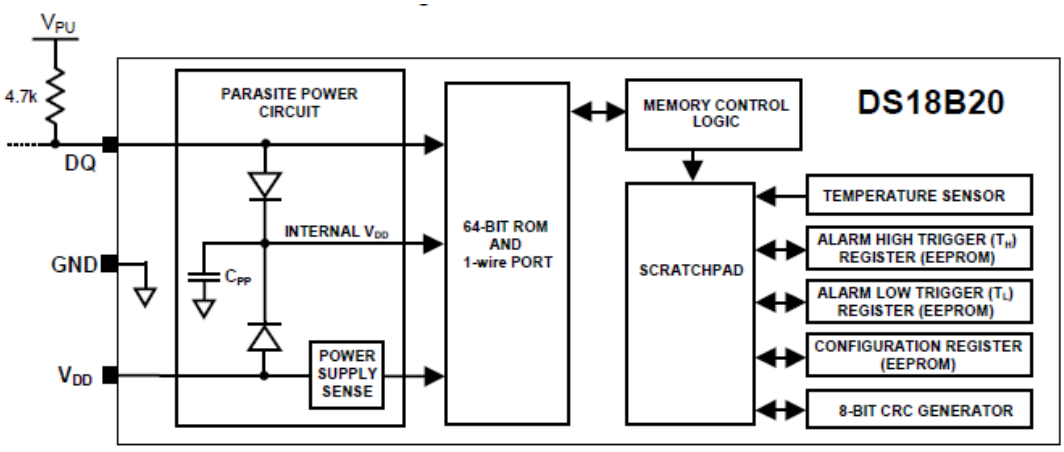
Základní parametry DS18B20:

- Napájecí napětí: 3 - 5.5V
- Rozsah měření: -55 - +125°C
- Přesnost ±0.5°C v rozsahu měření -10 - +85°C
- Doba převedení teploty na 12 bitové číslo: maximálně 750ms

2.1.1 Blokové schéma čidla DS18B20

Blokové schéma čidla je uvedeno na obrázku 2-1. Hlavní části čidla je 64 bitová paměť, kde je uložený unikátní kód čidla, který se používá při adresování čidla v případě, že je na sběrnici více podřízených zařízení. Další částí je paměťový blok, který je složen z registrů pro práci s čidlem. Paměťový blok obsahuje tyto registry:

- dvoubajtový teplotní registr, kde se ukládá digitální výstup z teplotního senzoru
- dva registry používané při funkci ALARM
- konfigurační registr – nastavení rozlišení převodu teploty do digitální podoby. Určuje, do kolika bitů bude uložená naměřená teplota, možnosti jsou: 9, 10, 11 nebo 12 bitů. Defaultně je čidlo nastaveno na rozlišení 12 bitů.

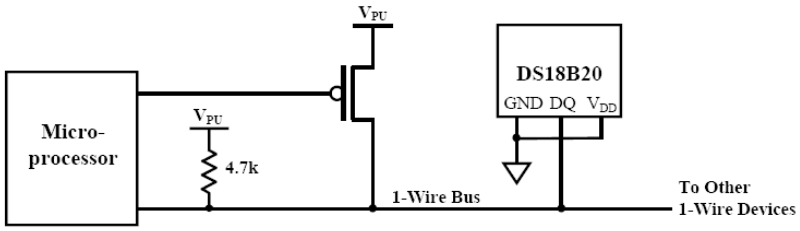


Obrázek 2-1: Blokové schéma čidla DS18B20 [1]

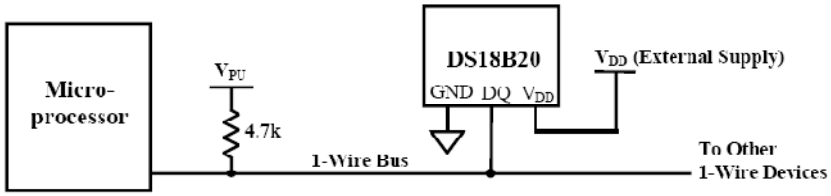
2.1.2 Zapojení čidla

Čidlo DS18B20 lze zapojit dvěma způsoby. První způsob je zapojení jen se dvěma vodiči viz Obrázek 2-2, v tom případě je pin VDD připojen na zem a čidlo je napájeno přímo z datového vodiče. Druhý způsob, který je použitý u teploměru, je zapojení čidla se třemi vodiči viz Obrázek 2-3.

V obou případech je datový pin DQ připojený na sběrnici. Sběrnice je přes zvedací rezistor připojena k napájení +5V, tím je zaručen klidový stav sběrnice - log. 1.



Obrázek 2-2: Zapojení čidla se dvěma vodiči [1]



Obrázek 2-3: Zapojení čidla se třemi vodiči [1]

2.1.3 Sběrnice 1-Wire

Komunikace mezi mikrokontrolerem a čidlem probíhá přes 1-Wire sběrnici. Na sběrnici je vždy jeden řídicí obvod-master (v případě teploměru mikrokontroler) a několik podřízených obvodů-slave (v případě teploměru čidlo DS18B20). Když je na sběrnici více podřízených obvodů (více teplotních čidel) musí se nejprve vybrat jeden obvod, se kterým chceme komunikovat. Výběr obvodu spočívá v tom, že známe jeho jedinečný 64 bitový kód, vysláním tohoto kódu zajistíme, že komunikovat budeme jen s jediným obvodem, jehož kód je shodný s vyslaným kódem. Ostatní obvody budou čekat na reset, podrobnější průběh komunikace bude uveden v kapitole 2.1.4. Sběrnice se skládá z jednoho datového vodiče a společné zemi. Logická hodnota na sběrnici je dána napětím mezi těmito dvěma vodiči. V klidovém stavu je mezi vodiči napětí 5V(log. 1) toto napětí je na datový vodič přivedeno přes zvedací (pull-up) rezistor s hodnotou 4,7kΩ. Uzemněním datového vodiče, docílíme úrovně napětí 0V mezi datovým vodičem a společnou zemí, což představuje logickou 0. Aby mohl komunikovat mikrokontroler přes sběrnici 1-Wire, napsal jsem si knihovnu (ds18b20.c), ve které jsou metody pro komunikaci.

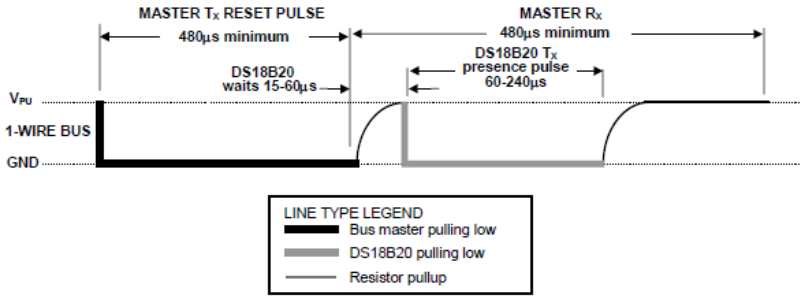
Základní komunikační prvky na sběrnici jsou:

- Inicializace
- Čtení bitu
- Zápis bitu

2.1.3.1 Inicializace

Inicializace se provádí vždy na začátku komunikace. Jde vlastně o kontrolu, zda je na sběrnici připojený podřízený obvod a jestli je připravený ke komunikaci. Skládá se ze dvou částí, kterými jsou reset pulz a prezenční pulz.

Reset pulz vysílá řídící obvod tím, že na sběrnici vyšle log. 0 na dobu minimálně 480µs, potom sběrnici uvolní a podřízený obvod by měl po 60µs vyslat log. 0 na dobu 60-240µs – to je prezenční pulz podřízeného obvodu, kterým odpověděl na reset pulz řídicího obvodu.



Obrázek 2-4: Průběh inicializačního pulzu na sběrnici 1-Wire [1]

Inicializaci u teploměru zajišťuje mikrokontroler, pomocí následující metody *dsInit*:

```
char dsInit(char dsBit)
{
	char ok=0;
	dsBus0(dsBit);
	_delay_us(490);
	dsBus1(dsBit);
	_delay_us(100);
	if((dsPin & (1<<dsBit))==0) ok=1;
	_delay_us(490);
	if((dsPin & (1<<dsBit))==0) ok=0;
	return ok;
}
```

Metoda *dsInit* provede reset pulz a počká 100µs, poté kontroluje, jestli byl na sběrnici vyslaný prezenční pulz. Takto lze zjistit, jestli je vůbec na sběrnici nějaký podřízený obvod připojen. Metoda vrací hodnotu 1, když byl vyslaný prezenční pulz v opačném případě, vrací hodnotu 0.

2.1.3.2 Zápis, čtení

Další důležité prvky pro komunikaci na sběrnici jsou zápis log. 1 a zápis log. 0. Data se zapisují na sběrnici ve slotech. Každý bit je obsažen v jednom slotu, jehož délka je minimálně 60µs. Začátek

každého slotu zahajuje vždy řídicí obvod, který vyšle na sběrnici log. 0 na dobu delší než je 1µs. Potom záleží, jestli bude řídící obvod data zapisovat nebo číst.

Pokud bude řídící obvod zapisovat, tak nechá na sběrnici log. 0 – tím zapíše bit s hodnotou 0, v opačném případě sběrnici uvolní, a přes pull-up rezistor se na datovou linku dostane log. 1 – zapíše bit s hodnotou 1.

Metoda *dsWriteBit* pro zápis bitu na sběrnici:

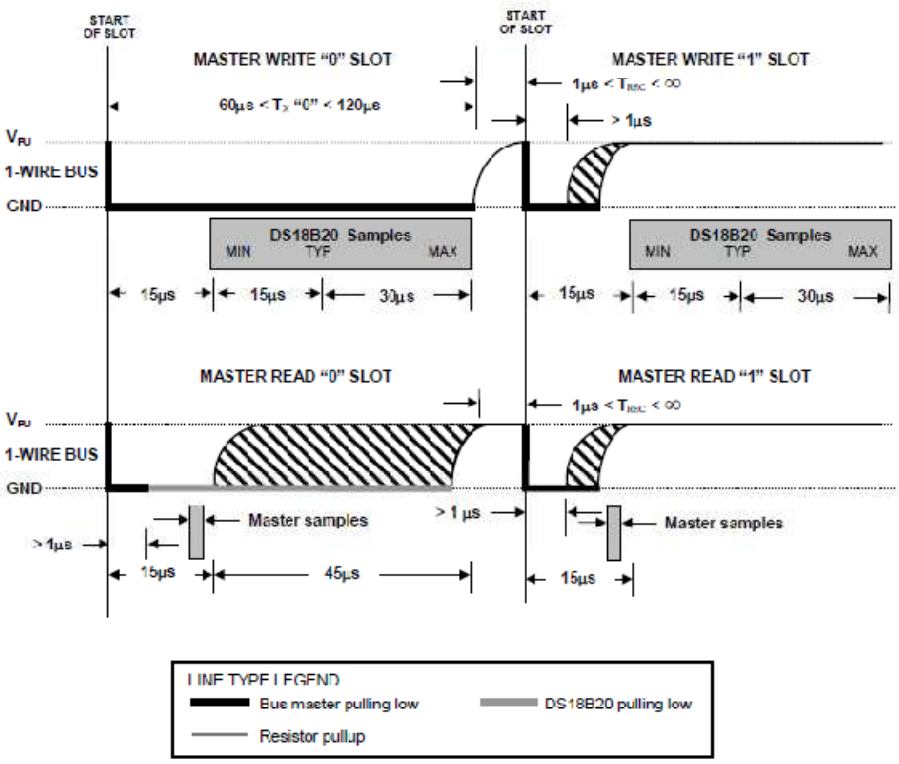
```
void dsWriteBit(char v, char dsBit)
{
	dsBus0(dsBit);
	_delay_us(10);
	if(v) dsBus1(dsBit);
	_delay_us(55);
	dsBus1(dsBit);
	_delay_us(5);
}
```

Pokud bude řídicí obvod číst, tak sběrnici uvolní a řízení na ní přebírá podřízený obvod, který nechá linku v log. 0 a řídicí obvod přečte bit s hodnotou 0 nebo sběrnici uvolní – log. 1 a řídicí obvod přečte bit s hodnotou 1.

Metoda *dsReadBit* pro čtení bitu na sběrnici:

```
char dsReadBit(char dsBit)
{
	char c=0;
	dsBus0(dsBit);
	_delay_us(5);
	dsBus1(dsBit);
	_delay_us(7);
	if((dsPin & (1<<dsBit))!=0) c=1;
	_delay_us(50);
	return c;
}
```

Časové průběhy zápisu a čtení jsou uvedeny na obrázku 2-5, podle těchto průběhu jsou napsané výše uvedené metody.



Obrázek 2-5: Časování zápisu/čtení jednoho bitu na sběrnici 1-Wire [1]

2.1.4 Komunikace s čidlem DS18B20

Komunikace s čidlem probíhá ve 3 fázích:

- Inicializace
- ROM Command
- DS18B20 Function Command

2.1.4.1 Inicializace

Všechny datové přenosy začínají inicializační sekvencí. Tato sekvence se skládá z reset pulzu, který vysílá řídicí obvod a prezenčního pulzu, který vysílá podřízený obvod. Vysláním prezenčního pulzu informuje řídicí obvod, že je na sběrnici a je připraveno ke komunikaci.

2.1.4.2 ROM Command

ROM Command je příkaz, který vysílá řídicí obvod po úspěšné inicializaci, kdy master zaregistroval prezenční pulz. Tento příkaz souvisí s unikátním 64 bitovým kódem každého podřízeného zařízení. Zde je popis příkazů, které jsou použity v teploměru:

- Search ROM F0h – pomocí toho příkazu lze zjistit adresy všech podřízených zařízení na sběrnici. Pomocí toho příkazu si teploměr prohledá sběrnice, zjisti a uloží si adresy všech připojených čidel.
- Match ROM 55h – tímto příkazem se vybírá jedno čidlo, se kterým chceme dále pracovat, po vyslání toho příkazu následuje 64 bitová adresa daného čidla.
- Skip ROM CCh – tento příkaz se hodí v případě, když je na sběrnici jedno čidlo se kterým chceme komunikovat nebo když chceme všem čidlům poslat stejný funkční příkaz. Po tomto příkazu se neposílá žádna adresa, ale následuje přímo funkční příkaz (Function Command).

2.1.4.3 DS18B20 Function Command

Function Command – příkaz, kterým říkáme čidlu, co má dělat. Zde jsou na výběr příkazy, kterými můžeme dávat povel k převedení teploty do digitální podoby, dále můžeme číst teplotu nebo můžeme měnit konfigurační registr a registry používané pro ALARM funkci. V teploměru jsou použity následující příkazy:

- Convert T 44h – po tomto příkazu začne čidlo převádět změřenou teplotu do dvou bajtových registrů MSB a LSB. Během převodu čidlo drží na sběrnici log. 0, dokud není převod dokončen, tím řídicí obvod zjisti, že je převod dokončen.
- Read scratchpad BEh – tento příkaz umožňuje řídicímu obvodu číst data z čidla. Čidlo je po tomto příkazu začne vysílat nejnižším bitem bajtu 0, řídicí obvod může kdykoliv vysílání zastavit vysláním reset pulzu.

2.2 Mikrokontroler Atmel ATmega8L

ATmega8L je 8 bitový mikrokontroler založený na AVR RISC architektuře.

Základní parametry:

- 8KB flash paměti programu
- 512B EEPROM
- 1KB SRAM
- programovatelný sériový USART
- 32 programovatelných vstupů/výstupů
- pracovní napětí: 2,7V – 5,5V
- pracovní frekvence: 0 – 8 MHz

Tento obvod je řídícím prvkem teploměru. Komunikuje s teplotními čidly, ovládá LCD displej a komunikuje s počítačem prostřednictvím převodníku FT232RL. Funkci tohoto obvodu zajišťuje firmware.

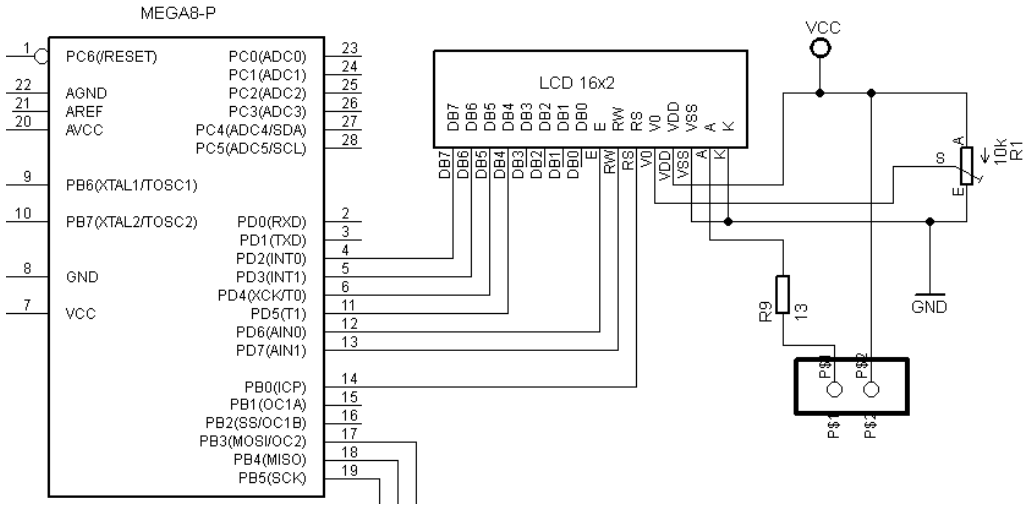
2.3 Zobrazovací jednotka

Jako zobrazovací jednotka je použitý LCD displej MC1602E-SYL/H. Informace o displeji jsou přebrány z datového listu [2].

Základní parametry:

- dvouřádkový displej se 16 znaky/řádek
- LED podsvícení
- LCD řadič: S6A0069

Displej je u teploměru použitý ve 4 bitovém režimu, zapojeny jsou čtyři datové vývody LCD displeje DB4 - DB7 a tři řídící vývody: E, RW, RS. Všechny tyto vývody jsou připojeny na port mikrokontroleru, přes který je displej řízen. Další vývody slouží k napájení displeje a napájení podsvícení displeje.



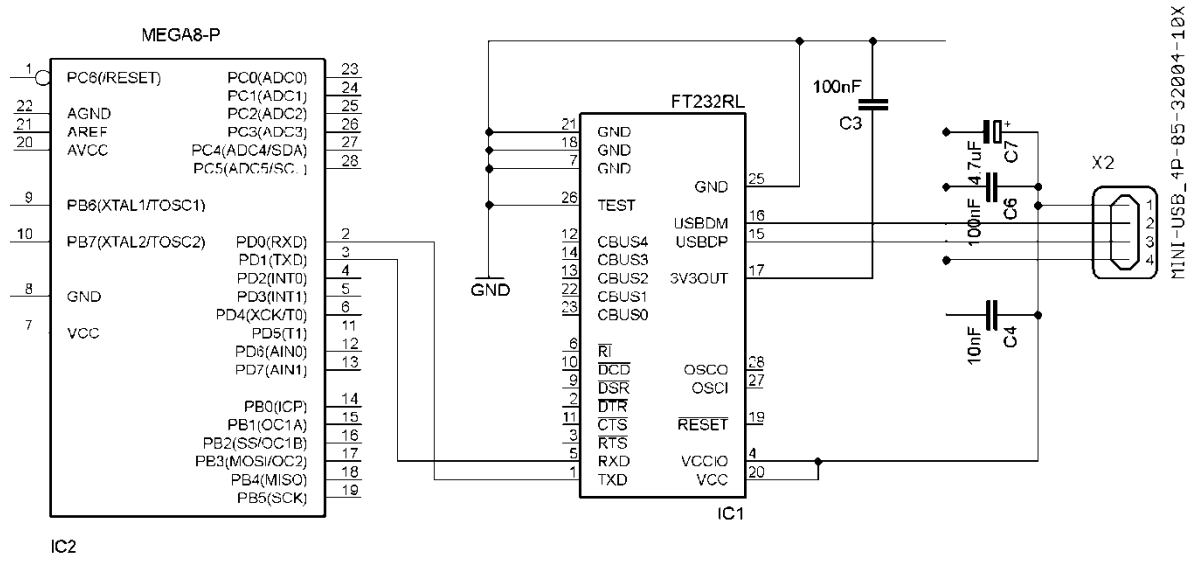
Obrázek 2-6: Schéma zapojení LCD displeje

Pro komunikaci s displejem byla vytvořena knihovna *lcd.c*, která zajišťuje základní operace s displejem. V knihovně je implementována metoda pro inicializaci displeje ve 4 bitovém režimu, dále jsou implementovány metody pro výpis textu, znaku a smazáni displeje. Informace o řízení displeje jsem čerpal ze zdroje [7], pomocí kterého jsem napsal metody pro komunikaci s displejem.

2.4 FTDI Chip FT232RL

Informace o tomto obvodě jsou čerpány z datového listu [3]. Obvod FT232RL zajišťuje převod komunikace mezi USB a UART rozhraními. Pomocí tohoto obvodu je zajištěna sériová komunikace mezi teploměrem a počítačem. Na straně teploměru je k převodníku připojený mikrokontroler Atmel ATmega8, ve kterém je obsažený UART (univerzální asynchronní přijímač vysílač). Zapojení obvodu

UART je realizováno pomocí dvou vodičů RXD a TXD. Pin RXD převodníku je připojený na pin TXD mikrokontroleru a pin TXD převodníku je připojen na RXD mikrokontroleru. Ze strany počítače je převodník připojený k USB, použity jsou piny USBDP a USBDM. Schéma zapojení je uvedeno na obrázku 2-7.



Obrázek 2-7: Schéma zapojení převodník

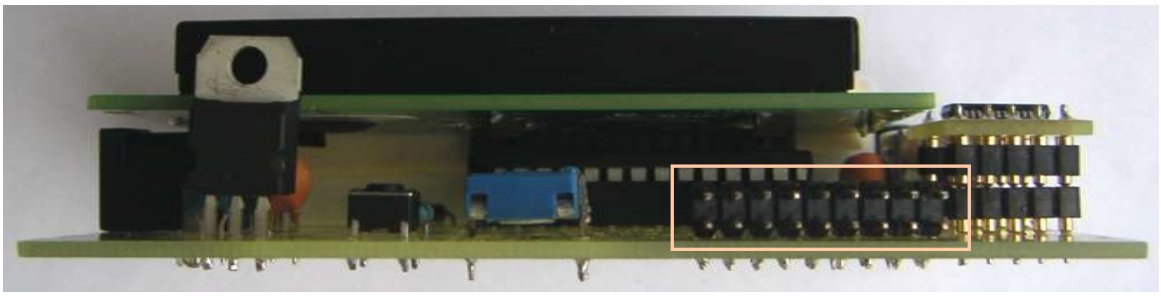
2.5 Popis kompletního zapojení

Kompletní schéma zapojení je uvedeno v příloze. Schéma se skládá z výše uvedených částí, jako je LCD displej, převodník FT232RL a teplotní čidla DS18B20. Napájení teploměru zajišťuje stabilizátor LM7805. Na vstup stabilizátoru stačí přivést napětí kolem 12V, maximální napětí uvedené v datovém listě součástky je 35V [5]. Na výstupu stabilizátoru je +5V proti zemi (GND). Zem je pro všechny součástky společná a +5V jsou napájený všechny součástky, až na výjimku převodníku FT232RL, který je napájen přímo z USB. K mikrokontroleru je připojen externí krystalový oscilátor s frekvencí 7.3728MHz, který je zdrojem hodinového signálu, výběrem této frekvence by měla být zajištěna nulová chyba při přenosu dat sériového portu při přenosové rychlosti: 9600 Baudů.

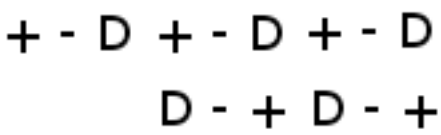
Po návrhu schématu zapojení následoval návrh plošného spoje. Plošný spoj jsem navrhoval ze schématu v PC aplikaci Eagle, ve které jsem také vytvářel schéma zapojení. Po návrhu desky plošného spoje jsem desku vytvořil foto cestou a napájel všechny potřebné součástky a konektory. Takto vznikl první prototyp digitálního teploměru (obrázek 2-10), pro který jsem následně vyvíjel firmware.

LCD displej je k DPS teploměru připojen přes pinovou lištu, na DPS je připájená lišta se 16 dutinkami a na LCD displej je připájena lišta se 16 piny. Podobně je řešený i obvod FR232RL, pro který jsem si navrhl zvláštní DPS s pinovými lištami a na DPS teploměru je pro ni připravená patice.

Pro připojení čidel k teploměru složí dvouřadý pinový konektor uvedený na obrázku 2-8 s popisem jednotlivých pinů na obrázku 2-9.



Obrázek 2-8: Pohled na konektor pro připojení čidel

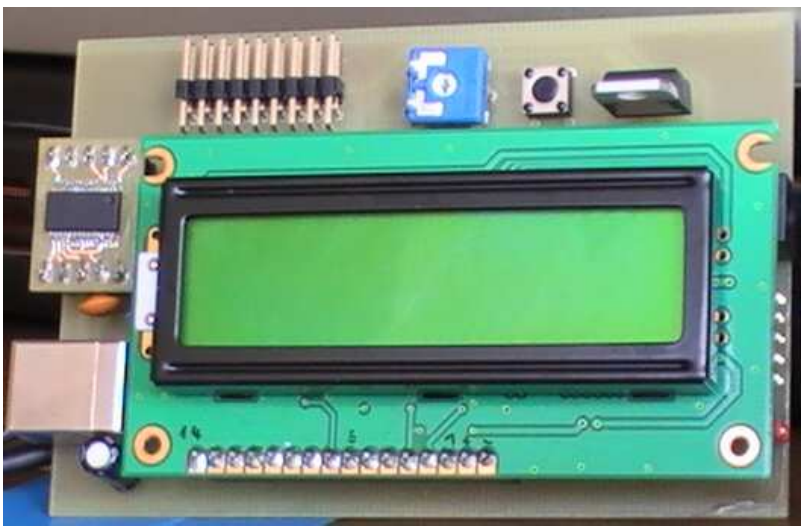


Obrázek 2-9: Popis pinů konektoru

Při pohledu na tento konektor jsou na obrázku 2-10 popsány piny pro připojení čidel:

Pin +5V, označení: +
Pin zem (GND), označení: -
Pin datové linky (sběrnice 1-Wire), označení: D

Počet datových linek na konektoru je pět, každá z nich představuje sběrnici 1-Wire, čidla lze libovolně připojit na kteroukoliv sběrnici (samozřejmostí je společná zem čidla a teploměru). Na jednu sběrnici lze připojit i více čidel, v tom případě jsou čidla zapojena paralelně.



Obrázek 2-10: Digitální teploměr

3 Firmware

Program pro mikrokontroler je napsaný v programovacím jazyce C v prostředí AVR Studio 4. Aby bylo možné v tomto prostředí programovat v jazyce C, je nejdříve nutné stáhnout balík WinAVR, což je sada programů umožňující vývoj firmware pro AVR mikrokontrolery pod operačním systémem Windows. Hlavní součásti balíku WinAVR je:

- AVR GNU Compiler Collection (GCC) 4.3.3
    - Překladač jazyka C pro AVR mikrokontrolery

- avr-libc 1.6.7cvs
    - Standardní knihovna jazyka C pro mikrokontrolery AVR

3.1 Implementace

3.1.1 Vlastní datové typy

Zde je uveden popis vlastních datových typů s definicí.

Datový typ *CIDLO* je struktura, která představuje v programu jedno fyzické čidlo. Položka *id* je jedinečné číslo přiřazené čidlu při nalezení jeho kódu na sběrnici. Dále je zde uložený 64 bitový kód daného čidla, pomocí kterého se čidlo adresuje na sběrnici. Další položkou ve struktuře je dvou bajtové pole pro ukládání teploty přečtené z teplotních čidel (registry MSB a LSB).

```
typedef struct{
        char id;
        char sn[8];
        char t[2];
}CIDLO;
```

Druhý datový typ je struktura *SBERNICE*. Tato struktura představuje jednu sběrnici 1-Wire. Skládá se z proměnné *počet*, která obsahuje informaci o tom, kolik je na sběrnici čidel a z ukazatele *cidla* na datový typ *CIDLO*. Je zde použitý ukazatel, protože není dopředu známo, kolik čidel bude ke sběrnici připojeno. Ukazatel se nastaví, až při inicializaci na dynamicky vytvořené pole typu *CIDLO*, kde počet prvků pole bude stejný jako počet připojených čidel.

```
typedef struct{
        char pocet;
        CIDLO *cidla;
}SBERNICE;
```

Posledním datovým typem je struktura *SKUPINA*. Tato struktura obsahuje informace pro práci se skupinami, ze kterých se poté počítá průměrná hodnota teploty. Prvky struktury jsou:

- *pocet* – definuje počet, z kolika čidel se bude počítat průměrná teplota

- **cidla* – ukazatel na pole typu char, prvky pole jsou pořadové čísla teplotních čidel, tak jak jsou uloženy v paměti. Pomocí těchto pořadových čidel se vyhledávají naměřené hodnoty daných čidel a počítá se z nich průměrná teplota.
- *kladna, pcela, pdes* – proměnné, do kterých se ukládá průměrná teplota, jsou zde informace o tom, zda je teplota kladná nebo záporná, její celá část a desetinná část
- *dalsi* – ukazatel ukazující na následující skupinu v pořadí, v případě že žádná další skupina neexistuje je nastaven na hodnotu NULL

```
typedef struct skupina{
        char pocet;
        char *cidla;
        char kladna;
        char pcela;
        char pdes;
        struct skupina *dalsi;
}SKUPINA;
```

3.1.2 Globální proměnné

char dsDesetiny[] = {0,1,1,2,2,3,4,4,5,6,6,7,7,8,9,9};

Teplotní čidlo DS18B20 je schopno měřit s rozlišením 1/16 °C, to představuje hodnotu se 4 desetinnými místy a krokem 0,0625 ºC, když se vezme v úvahu kalibrační přesnost čidla ±0,5°C v rozsahu teplot -10 až +85 ºC, jsou 4 desetinné místa zbytečná a postačí jen jedno, hodnoty desetinných míst jsou uloženy v globální proměnné – pole: *dsDesetiny* podle následující tabulky:

0	0	0
1	0,0625	0
10	0,125	1
11	0,1875	2
100	0,25	3
101	0,3125	3
110	0,375	4
111	0,4375	4
1000	0,5	5
1001	0,5625	6
1010	0,625	6
1011	0,6875	7
1100	0,75	8
1101	0,8125	8
1110	0,875	9
1111	0,9375	9

Tabulka 3-1: Převod desetinné části teploty

První sloupec tabulky 3-1 obsahuje binární hodnoty jak je desetinná část uložena v teplotním čidle. Druhý sloupec je už hodnota odpovídající desetinné části teplotě a třetí sloupec tabulky obsahuje zaokrouhlení hodnot druhého sloupce v desetinách, protože by bylo zbytečně pracovat s více desetinnými místy, postačí pouze jedno, jehož hodnota je v třetím sloupci.

SBERNICE sber[5];

Teploměr umožňuje připojení teplotních čidel až k 5 sběrnicím 1-Wire, které jsou realizovány mikrokontrolerem – port C piny 1 až 5. Protože je možné na jednu sběrnici připojit více čidel, které se musí poté adresovat, je třeba někde mít uložené kódy čidel. Právě k tomu slouží globální proměnná: *sber*. Je to pole 5 prvků typu *SBERNICE*, ke každé fyzické sběrnici je se používá jeden prvek z tohoto pole. Jednotlivé prvky pole obsahují informaci o počtu čidel na fyzické sběrnici (*pocet*) a ukazatel na dynamicky vytvořené pole (*cidla*) typu *CIDLO*, obsahující informace o jednotlivých čidlech. Pomocí této proměnné můžeme procházet všechny čidla, uchovávat a načítat jejich teploty pro pozdější práci s nimi (počítání průměrných hodnot, zobrazení na LCD displeji, posílání dat sériovou linkou).

char cidlaLCD[]={1,0,0,0,0,0};

Proměnná *cidlaLCD* je pole, které nám říká, co se bude zobrazovat na LCD displeji. Skládá se ze šesti hodnot:

- 1. hodnota určuje počet zobrazovaných teplot jednotlivých čidel
- 2. hodnota určuje počet zobrazovaných průměrných hodnot
- Za druhou hodnotou následují pořadové čísla teplotních čidel, jejichž teplota se bude zobrazovat, počet pořadových čísel odpovídá první hodnotě v poli. Hned za pořadovými čísly čidel následují pořadové čísla identifikující průměrovou skupinu.

Teploměr je navržený pro zobrazení čtyř teplot současně. Pokud bychom chtěli zobrazit více teplot na LCD displeji, teploty by nebyly zobrazeny celé nebo by zobrazení bylo nepřehledné. Pro jedu teplotu je vyhrazeno osm znaků na LCD displeji.

Pole *cidlaLCD* je po spuštění programu nastaveno tak, jak je uvedeno výše. Hodnoty tohoto pole jsou nastaveny v průběhu inicializace dle počtu připojených čidel nebo lze jednoduše změnit pomocí PC aplikace přes grafické rozhraní.

SKUPINA *skupiny;
char pocSkupin;

Další globální proměnná: *skupiny* odkazuje na první skupinu čidel. Po spuštění programu je automaticky vytvořena skupina z prvních dvou čidel, v případě že jsou dvě a více čidla připojena. V opačném případě je ukazatel nastaven na hodnotu *NULL* a další proměnná: *pocSkupin*, která se skupinami souvisí je nastavena na hodnotu: 0. Proměnná *pocSkupin* uchovává aktuální počet definovaných skupin.

unsigned char buffer[16];
char iBuffer=0;
char noveData=0;

Poslední tři globální proměnné realizují prostor pro uložení přijatých dat ze sériového port. Prostor je realizovaný šestnácti bajtovým polem *buffer*, do kterého se ukládají data přijata ze sériového portu. Proměnná *iBuffer* je aktuální index pole – ukazuje, na kterou pozici v poli *buffer* se zapíši nová data. Proměnná *noveData* se v případě příjmu nových dat nastaví na hodnotu 1 jako informace, že jsou v poli *buffe*r nová data a čekají na zpracování. Popis přijetí a zpracování dat je uveden později v kapitole 3.1.10.

3.1.3 Hlavní program

Program se skládá z následujících částí:

- Inicializace
- Přečtení teplot z čidel
- Výpočet průměrných teplot
- Zobrazení teplot na LCD displeji
- Odeslání dat
- Zpracování přijatých dat

Inicializace se provede jen jednou po spuštění programu, ostatní části se vykonávají v nekonečné smyčce.

3.1.4 Inicializace

V první části inicializace se nastavuje mikrokontroler, protože jeho registry jsou nastaveny na defaultní hodnoty, které se nastaví vždy po připojení napájení mikrokontroleru nebo při resetu. Porty mikrokontroleru jsou defaultně nastaveny jako vstupní. V programu se nastaví jako výstupní porty celý port D nultý pin portu B, tyto porty jsou připojeny k LCD displeji a přes ně se displej řídí (schéma zapojení je na obrázku 2-6).

Potom se u mikrokontroleru musí nastavit a povolit sériový port, zde se nastavuje rychlost přenosu dat, počet datových bitů a počet stop bitů. Sériový port je nastavený na rychlost 9600 Baudu, 8 datových bitů a 2 stop bity metodou *USART_Init*. Piny RX a TX mikrokontroleru jsou standardně nastaveny jako digitální vstup/výstup. Povolením sériového portu jsou aktivovány alternativní funkce těchto pinů: příjem a odesíláni dat v rámci USART. Piny jsou připojeny k převodníku FT232RL (schéma zapojení je na obrázku 2-7).

Dále se inicializuje LCD displej pomocí metody *LCDinit4b()*. V metodě se nastavuje, že komunikace bude probíhat, po čtyřech datových vodičích, displej bude zobrazovat na dvou řádcích, displej se zapne, kurzor se nebude zobrazovat a displej se smaže. Po inicializaci displeje je na něm vypsán text *Teplomer* po dobu dvou vteřin.

Dále je třeba zjistit počet všech připojených čidel na jednotlivých sběrnicích a uložit jejich 64 bitové kódy. Toto provádí metoda *cidlaInit()*, která postupně prochází všechny sběrnice – Port C, piny 1 až 5. Na každé sběrnici se nejprve spočítají čidla metodou *char dsPocCidel(char dsBit, char uloz)*, která

vrací počet nalezených čidel. Parametry předávané této metodě jsou pin sběrnice a hodnota *uloz* – tím se řídí, jestli se čidla jen počítají nebo se zároveň ukládají jejich kódy (hodnota 0 – čidla se jen počítají, hodnota 1 – ukládají se i kódy čidel). Když je známý počet čidel na sběrnici může se dynamicky alokovat prostor v paměti pro daný počet čidel:

```
CIDLO *c=(CIDLO*)malloc(pocet*sizeof(CIDLO));
```

V případě, že výsledkem alokace není hodnota NULL, hodnota ukazatele *c* se tedy přiřadí položce *cidla* globální proměnné *sber* aktuálně prohledávané sběrnice. Poté opět zavolá metoda *dsPocetCIdel* na stejné sběrnici, ale tentokrát s hodnotou *uloz*=1. Tím se sběrnice prohledá znovu, ale už se ukládají kódy čidel.

```
if(c!=NULL)
{
        sber[s-1].cidla=c;
        dsPocCidel(s,1);
 }
```

Dalším krokem je inicializace průměrové skupiny. Pokud je k teploměru připojeno více než jedno čidlo, tak je automaticky vytvořena průměrová skupina P0, které počítá průměrnou teplotu z prvních dvou nalezených čidel. To se provede v metodě *prumerInit().*

Metoda *prumerInit* dále nastavuje hodnoty globálního pole *cidlaLCD*, kde podle počtu připojených čidel nastaví, které teploty čidel budou zobrazovány na LCD displeji. V případě, že není připojeno žádné čidlo, nastaví první pozici pole *cidlaLCD* hodnotu nula, tím říká, že nebude zobrazována žádna teplota. V případě, že je připojeno k teploměru více než jedno čidel, nastaví hodnoty pole takto:

```
cidlaLCD[0]=2;
cidlaLCD[1]=1;
cidlaLCD[3]=1;
```

Tím se nastaví, že se budou zobrazovat teploty prvních dvou nalezených čidel a průměrná teplota z těchto dvou čidel.

Na konci inicializaci se ještě povolí globální přerušení metodou *sei().* U teploměru může přerušení vyvolat jen přijetí dat sériového portu, protože tyto data je nutné hned číst a uložit.

3.1.5 Hledání čidel na sběrnici

Abychom mohli pracovat s jednotlivými čidly na sběrnici, je potřeba zjistit unikátní kódy všech čidel na dané sběrnici. Postup při hledání čidel je následující: na sběrnici se vyšle inicializační pulz, po něm následuje ROM příkaz 0xF0, který říká všem čidlům na sběrnici, aby vysílali postupně bity svého unikátního kódu. Jeden bit kódu se vysílají čidla dvakrát, jednou jeho přímou hodnota a jednu negovanou hodnota. Tady mohou nastat tři stavy, podle toho co bylo na sběrnici přečteno:

- Bity 0 a 1 – hodnota bitu všech čidel je 0
- Bity 1 a 0 – hodnota bitu všech čidel je 1

- Bity 0 a 0 – v tomto případě nastal konflikt, bit některých čidel je 0 a některých 1

Po přečtení dvou bitů na sběrnici musí řídící obvod vyslat bit s hodnotou odpovídající hodnotě prvního ze dvou přijatých bitů. V případě konfliktu je na řídicím obvodu co pošle za hodnotu a tím zvolí čidla, se kterými chce dále komunikovat. Čidla, jejichž hodnota bitu, není shodná s hodnotu bitu vyslaného řídicím obvodem, již dále na sběrnici nekomunikují, dokud není vyslán reset pulz.

Princip hledání čidel tedy spočívá v tom, že v případě když nastane konflikt tak řídicí obvod vyšle hodnotu bitu 0 a zapamatuje si pozici bitu, kdy došlo ke konfliktu. Takto se pokračuje, dokud není přečteno 64 bitů kódu jednoho čidla. Všechny vyslané bity kódu se ukládají a jsou dále využity při hledání dalších čidel. Když si pamatujeme pozici, kdy došlo k poslednímu konfliktu, můžeme se v dalším průchodu na tuto pozici zase dostat (odesíláním stejných bitů jako v předchozím případě až po pozici posledního konfliktu) a vyslat opačnou hodnotu než v předchozím průchodu všemi bity. Tím budeme komunikovat s jiným čidlem než v předchozím případě. Takto se prochází celá sběrnice pořád dokola, dokud se nenastane případ, že by konflikt nenastal (v tom případě jsme prošli všechny čidla na sběrnici a nemá dále smysl pokračovat). Přesný popis vyhledání čidel je uveden ve zdroji [8], podle tohoto zdroje je napsaná metoda: *dsPocCidel*, která pomocí popsaného algoritmu prohledává sběrnice.

3.1.6 Přečtení teplot

Zde se program nachází na začátku nekonečné smyčky a je nutné zjistit všechny teploty čidel. Čtení teplot čidel je zajištěno metodou *nactiTeploty(),* která postupně prochází sběrnice a na každé z nich vyšle inicializační impulz, po něm se následuje ROM příkaz 0xCCh – říká, že se nebude adresovat a následně se vyšle funkční příkaz 0x44h všem čidlům na sběrnici. Všechny čidla převádějí naměřenou teplotu do digitální podoby s následným uložením do své paměti (registry MSB a LSB), v době převodu je sběrnice v log. 0 a čeká se, až všechny čidla budou mít teplotu převedenou. Potom se postupně prochází jednotlivá čidla: na sběrnici je vyslán inicializační impulz, ROM příkaz 0x55h – bude se adresovat, následuje adresa čidla, funkční příkaz 0xBEh a následně čidlo začne vysílat obsah registrů: LSB a MSB, které jsou uloženy. Po přečtení dvou bajtů je odesílání dalších bajtů ukončeno reset pulzem a pokračuje se s načítáním teplot dalších čidel stejným postupem.

3.1.7 Výpočet průměrných teplot

Průměrné teploty se počítají v metodě *prumer()*. Pro výpočet průměrných teplot se postupně musí projít všechny definované skupiny a v každé skupině se projdou všechny čidla, ze kterých se průměr počítá. K procházení skupin se používá ukazatel na datový typ *SKUPINA*, na začátku je nastaven na první skupinu. Celkový počet skupin je uložen v proměnné: *pocSkupin*. Jestli je nějaká další skupina dostaneme se k ní přes aktuální skupinu položkou *dalsi* ve struktuře datového typu *SKUPINA*, což je ukazatel na následující skupinu (*s=s->dalsi;*). V každé skupině je definován počet čidel ve skupině (položka struktury - *pocet*) a ukazatel na pořadová čísla čidel, které tvoří skupinu (položka struktury - *cidla*). Uloženou teplotu čidla dostaneme pomocí metody: *void zjistiTeplotu(char idh, char *cela, char *des, char *kladna);* této metodě musíme poskytnout pořadové číslo čidla (*idh*) a adresy tří proměnných do kterých metoda uloží teplotu (*cela, des, kladna*).

```
char idh = s->cidla[c];
```

```
char cela;
char des;
char kladna;
zjistiTeplotu(idh, &cela, &des, &kladna);
```

Před začátkem procházení jednotlivých čidel skupiny je definována proměnná *prumer* typu int. K této proměnné postupně při procházení čidel, přičítám nebo odečítám teplotu čidla, na základě jestli byla kladná nebo záporná.

```
if(kladna==1) prumer+=cela*10+des;
else prumer-=(cela*10+des);
```

Protože teploměr pracuje i s desetinami teplot, mohl by se použít datový typ double nebo float pro práci s reálnými čísly, práce s takovými datovými typy by byla v 8 bitovém mikrokontroleru neefektivní (při výpočtech s proměnnými datových typu real nebo double je využita velká část paměti programu) a proto se v teploměru používá datový typ int a pracuje se s deseti - násobkem teploty. Celá část se vynásobí deseti a k ní se přičte desetinná část. Když jsou sečteny všechny 10 násobky teplot všech čidel skupiny, proměnné *prumer* se vydělí počtem čidel ve skupině.

```
prumer=prumer/pocCidel;
```

Výsledná průměrná teplota se ukládá zase do proměnných: *kladna, pcela, pdes*. Proto je potřeba nejprve nastavit proměnnou kladna, jednoduchým porovnáním jestli průměr je menší než nula.

```
if(prumer<0)
{
        s->kladna=0;
        prumer=prumer*(-1);
}
else s->kladna=1;
```

V případě že průměr byl záporný, vynásobíme ho -1, abychom dostali kladnou hodnotu, protože s průměrem budeme dále počítat a je potřeba abychom se pohybovali v kladných číslech.

Výsledná celá část průměrné teploty bude tedy obsah proměnné: *prumer* a protože jsem pracovali s deseti násobkem teploty, musíme ji zase deseti vydělit. Výslednou hodnotu uložíme do proměnné skupiny: *pcela*. Desetinou část získáme jako zbytek po dělení průměru 10, výsledná desetinná část průměru se uložíme do proměnné: *pdes.*

```
s->pcela=(prumer/10);
s->pdes=(prumer%10);
```

3.1.8 Zobrazení teplot na LCD displeji

Teploty na LCD displeji zobrazí metoda *zobrazTeploty()*, tato metoda pracuje s globální proměnnou *cidlaLCD*. Metoda první přečte hodnotu pole *teplotyLCD* na první pozici (*teplotyLCD[0]*), tím zjisti

kolik teplot, jednotlivých čidel bude zobrazovat. Nyní následuje smyčka, která se provede tolikrát, kolik je počet zobrazovaných teplot. Ve smyčce se zjisti pořadové číslo čidla (promenná *idh*), jehož teplota se bude zobrazovat.

```
char idh=cidlaLCD[2+i];
```

Dále definujeme 3 proměnné: *kladna, cela, des* a zavoláme metodu*: zjistiTeplotu(idh, &cela, &des, &kladna)*. Metoda zjistí pozici čidla, převede jeho teplotu a uloží do tří proměnných (*kladna, cela, des*). Po zjištění teploty se na displeji zobrazí znak *T* a za ním hned pořadové číslo čidla (např. T0), aby bylo jasné, jaká teplota se zobrazuje. Znaky na displej se vypisují pomocí metody: *LCDzna(char znak)*.

```
LCDznak('T');
LCDznak(idh+'0');
```

Výpis znaku *T* je jednoduchý, metodě předáme ascii hodnotu daného znaku. V případě vypsání proměnné obsahující jednociferné číslo se ascii hodnota čísla v proměnné dostane přičtením ascii hodnoty znaku nula.

Za výpisem znaku T a pořadového čísla čidla následuje už jeho naměřená teplota, tuto teplotu vypíše metoda *zobrazTeplotu*, které předáme tři zmiňované proměnné.

```
zobrazTeplotu(cela, des, kladna);
```

Po výpisu teploty na LCD displej je třeba zajistit, aby se další zobrazovaná teplota nezobrazila hned za předchozí teplotu, protože by výpis byl nepřehledný. Dalo by se to řešit vypsáním dalšího znaku: mezery. Tím bychom neměli kontrolu nad tím, jestli není třeba vypsat teplotu na další řádek displeje. Proto je zde použita proměnná: *poz*, která na při volání metody *zobrazTeplotu* nastavena na hodnotu 0 (vypisovat se bude na první řádek, první pozici displeje). Po zapsání jedné teploty na displej se hodnota proměnné *poz* zvýši o hodnotu 8 a displeji se pošle příkaz, aby se příště na tu pozici začalo zapisovat. Při zvyšování o hodnotu 8 se po druhém zvýšení (již jsou vypsány dvě teploty na prvním řádku) hodnota proměnné *poz* bude rovnat 16. To se po zvýšení proměnné kontroluje a v případě že dosáhne této hodnoty je třeba LCD displeji říct, aby se přesunul na druhý řádek. To se provede tak že se do proměnné *poz* vloží hodnota 0x40h a následně se displeji tato informace sdělí metodou *LCDpoz(poz).*

```
poz=poz+8;
if(poz==16) poz=0x40;
LCDpoz(poz);
```

Po zobrazení všech teplot následuje zobrazení průměrných teplot. Průběh je zde podobný jako při zobrazování teplot jednotlivých čidel, až na výjimku, že procházíme skupiny a po nalezení skupiny, jejíž průměrnou teplotu chceme zobrazit, nemusíme volat metodu *zjistiTeplotu*, ale hodnoty si načteme přímo z položek struktury datového typu *SKUPINA* (položky *pcela, pdes, pkladna*). Tyto hodnoty předáme metodě *zobrazTeplotu* jako v předcházejícím případě. Další změna je v tom, že teplota skupiny je na displeji zobrazena se znakem P (průměrná teplota) a pořadovým číslem skupiny (např. P0).

3.1.8.1 Popis jednotlivých metod

void zjistiTeplotu(char idh, char *cela, char *des, char *kladna)

Abychom mohli pracovat s teplotu příslušného čidla, je potřeba nejdříve zjistit jeho pozici v poli, poté načíst jeho teplotu a převést ji do příslušného tvaru (celá část, desetinná část a informace jestli je teplota záporná nebo kladná). To provede metoda *zjistiTeplotu*, které předáme pořadové číslo hledaného čidla (*idh*) a tři ukazatele na výsledné proměnné do kterých se teplota uloží (*cela, des, kladna*). Ke všem čidlům se dostaneme přes globální pole *sber* (představuje jednotlivé fyzické sběrnice), ale potřebujeme znát, na které sběrnici je čidlo připojené (*iSber*) a dále také pozici čidla (*iCidla*) v poli mezi ostatními čidly v případě, že jich je na sběrnici více. Pořadové čísla čidel se přidělují od nuly při inicializaci, tak můžeme pozici čidla dopočítat.

V poli sběrnic (*sber*) je v jednotlivých sběrnicích uložený počet čidel na sběrnici (*sber[i].pocet*). Na začátku si definujeme proměnnou *poc* s hodnotou: 0 a postupně procházíme všech pět sběrnic a do proměnné *poc* postupně přičítáme počet čidel na jednotlivých sběrnicích. Navíc si pamatujeme součet čidel na všech předchozích sběrnicích (proměnná *pred*). Dále kontrolujeme porovnáním, jestli součet všech čidel (*poc*) nepřesáhl hodnotu pořadového čísla hledaného čidla (*idh*).

```
char poc=0;
for(char s=0;s<5;s++)    {
        char pred=poc;
        poc=poc+sber[s].pocet;
        if(poc>idh)
```

Pokud součet přesáhne hodnotu *idh* víme, že na aktuální sběrnici je připojené hledané čidlo. Tím jsme se dostali ke sběrnici, na které je čidlo připojené a ještě potřebujeme zjistit pozici v poli mezi ostatními čidly. Pozici vypočteme tak, že od pořadového čísla odečteme součet všech čidel na předchozích sběrnicích (*pred*).

```
iSber=s;
iCidla=idh-pred;
```

Nyní známe pozici čidla (*iSber, iCidla*) a můžeme přečíst jeho poslední naměřenou teplotu, která byla uložená v předchozí metodě *nactiTeploty()*.

```
char lsb=sber[iSber].cidla[iCidla].t[0];
char msb=sber[iSber].cidla[iCidla].t[1];
```

Na konci metody zavoláme další metodu: *teplota(msb, lsb, cela, des, kladna)*, která převede tyto dva bajty (*msb, lsb*) z formátu získaného z čidla na formát se kterým se bude lépe pracovat (celá část teploty, desetinná část a informace jestli je teplota kladné nebo záporná).

void teplota(char dsMSB, char dsLSB, char *cela, char *des, char *kladna)

Teplotní čidlo ukládá naměřenou teplotu do dvou registrů (msb a lsb) ve své paměti (scratchpad). Hodnoty těchto dvou registru poté posílá po sběrnici 1-Wire. Protože by se s těmi dvěma registry špatně pracovalo, převedeme teplotu do lepšího formátu, který se bude skládat z informace, zda je teplota kladná nebo záporná, z celé části teploty a z desetinné části. Metodě *teplota* jsou předány

hodnoty: *dsMSB, dsLSB* a tři ukazatelé na proměnné, kde se uloží teplota ve zmiňovaném formátu (kladná/záporná, celá část, desetinná část teploty). Zde je ukázka formátu registrů MSB a LSB s příkladem hodnot těchto registrů a příslušnými teplotami:

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
LS Byte	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}
	bit 15	**bit 14**	**bit 13**	**bit 12**	**bit 11**	**bit 10**	**bit 9**	**bit 8**
MS Byte	S	S	S	S	S	2^6	2^5	2^4

Obrázek 3-1: Formát uložené teploty v registrech LSB a MSB [1]

TEMPERATURE	DIGITAL OUTPUT (Binary)	DIGITAL OUTPUT (Hex)
+125°C	0000 0111 1101 0000	07D0h
+85°C*	0000 0101 0101 0000	0550h
+25.0625°C	0000 0001 1001 0001	0191h
+10.125°C	0000 0000 1010 0010	00A2h
+0.5°C	0000 0000 0000 1000	0008h
0°C	0000 0000 0000 0000	0000h
-0.5°C	1111 1111 1111 1000	FFF8h
-10.125°C	1111 1111 0101 1110	FF5Eh
-25.0625°C	1111 1110 0110 1111	FE6Fh
-55°C	1111 1100 1001 0000	FC90h

*The power-on reset value of the temperature register is +85°C

Obrázek 3-2: Příklad některých teplot a jejich hodnot [1]

Nejprve se kontroluje, zda je teplota kladná nebo záporná. Tato informace je obsažena v MSB registru v pěti nejvyšších bitech jsou to znaménkové bity a jejich hodnota je stejná. Jsou-li bity jedničkové, je teplota záporná v opačném případě kladná.

```
if ((dsMSB & 0xf8) == 0) *kladna = 1;
else *kladna = 0;
```

Celá část teploty je uložena ve třech nejnižších bitech registru MSB a ve čtyřech nejvyšších bitech registru LSB. Hodnota celé části teploty odpovídá skutečnosti v případě, že je teplota kladná.
V případě záporné teploty je třeba od hodnoty: 0x7fh odečíst hodnotu celé části v registrech MSB a LSB, protože záporná hodnota teploty nejblíže nule je zastoupena nevyšší hodnotou v registrech MSB a LSB (hodnotou 0x7fh).

```
if(*kladna) *cela = (char)(((dsMSB << 4) & 0x70) | ((dsLSB >> 4)));
else *cela = 0x7f-(char)(((dsMSB << 4) & 0x70) | ((dsLSB >> 4)));
```

Desetinná část teploty je uložena v nejnižších čtyřech bitech registru LSB. Hodnota desetinné části je uložená ve čtyřech bitech, její hodnota je tedy 0 až 15. V případě kladné teploty stačí čtyřbitovou hodnotu převést pomocí tabulky, která je obsažena v globální proměnné *dsDesetiny*. V případě záporné teploty je potřeba nejprve zjistit jestli hodnota není rovna nule, v tom případě je třeba zvýšit hodnotu celé části o 1 a desetinná část teploty se rovná nule, v opačném případě je třeba odečíst od

hodnoty 16 čtyřbitovou hodnotu obsaženou v registru LSB. Následně převedeme desetinnou část pomocí globálního pole *dsDesetiny*.

```
if(*kladna) *des=(dsLSB & 0x0f);
else
{
        *des=(dsLSB & 0x0f);
        if (*des==0) *cela++;
        else *des=16-*des;
}
*des=dsDesetiny[*des];
```

void zobrazTeplotu(char cela, char des, char kladna)

Vypisovaná teplota se skládá ze znaménka teploty (kladná +, záporná -), celé části teploty (můžou to být až 3 znaky, když je teplota nad 100°C), desetinné tečky a jednoho znaku představující desetinnou část.

Celá část teploty se musí rozdělit na jednotlivé číslovky. Stovky se získají vydělením celé části stem, desítky se získají vydělením desíti zbytku po dělení celé části stem. Jednotky se zjisti jako zbytek po dělení desíti zbytku celé části vydělené stem.

```
char dig1=(cela/100);
char x=cela%100;
char dig2=(x/10);
char dig3=(x%10);
char dig4=des;
```

Na displej se nejprve vypíše znak plus nebo mínus podle hodnoty v proměnné kladna, když je hodnota proměnné rovna jedné vypíše se plus, jinak se vypíše mínus.

```
if (kladna) LCDznak('+');
else LCDznak('-');
```

Za znaménkem se vypíšou jednotlivé číslovky celé části teploty, desetinná tečka a desetinné část teploty. Ještě se zde kontroluje, jestli není první nebo první a zároveň druhá číslovka celé části teploty nulová, aby se zbytečně nevypisovala hodnota teploty s nulou na začátku (místo 023.2 se vypíše jenom 23.2 nebo v případě 002.2 se vypíše jen 2.2).

```
if(dig1!=0) LCDznak(dig1+'0');
if(dig1!=0 && dig2!=0) LCDznak(dig2+'0');
LCDznak(dig3+'0');
LCDznak('.');
LCDznak(dig4+'0');
```

3.1.9 Odeslání dat

Teploměr poskytuje pomocí sériové linky všechny důležité informace. Jsou to hlavně naměřené teploty jednotlivých čidel, průměrné teploty definovaných skupin čidel a nastavení toho co se zobrazuje na LCD displeji.

Všechny data se posílají v blocích. Jeden blok je složený z následujících části:

- 8 - bitová adresa cílového zařízení (pro PC jsem zvolil hodnotu 250)
- 8 - bitová adresa odesílatele (pro teploměr jsem zvolil hodnotu 150)
- 16 - bitový řídící příkaz – určuje, jaká informace se posílá v daném bloku
- 8 – bitová délka dat
- data – posílané informace, počet bajtů dat je stejný jako udána délka dat
- 16 – bitový cyklický redundantní součet dat (CRC) počítaný z dat

O vysílání všech dat v těchto blocích se stará metoda *posliData()*. Tato metoda posílá tři typy zpráv. Jsou to zprávy nesoucí informace o sběrnicích a čidlech na nich připojených, informace o zobrazovaných teplotách na LCD displeji a informace o definovaných skupinách.

Informace o sběrnicích a čidlech jsou definovány řídícím příkazem s hodnotou: 0x0101. Pro každé čidlo se posílá jeden blok o délce dat pět bajtů. Data této informace jsou složeny z:

- čísla sběrnice, na které je čidlo připojené
- počtu všech čidel na sběrnici
- pořadovým číslem čidla
- dva bajty teploty čidla ve formátu jakém teplotu ukládá čidlo (MSB, LSB)

Informace o zobrazovaných teplotách je definována řídícím příkazem s hodnotou: 0x0202. Délka dat je šest bajtů a jako data se posílají prvky globálního pole *cidlaLCD*, ve kterém jsou uloženy informace o zobrazovaných teplotách na LCD displeji.

Informace o skupinách je definována řídicím příkazem s hodnotou: 0x0303. Délka dat v této informaci je dána počtem čidel ve skupině sečteným s hodnotou pět. Data jsou složená z:

- pořadového čísla skupiny
- počtu čidel ve skupině
- pořadových čísel čidel ve skupině
- tří bajtů popisující hodnotu průměrné teploty skupiny (jestli je teplota kladná nebo záporná, celou část teploty a desetinnou část teploty)

V každém bloku se počítá 16 bitové CRC dat pomocí metody: *uint16_t _crc16_update(uint16_t __crc, uint8_t __data)* implementované v AVR Libc, CRC se počítá podle polynomu $X^{16}+X^{15}+X^2+1$ (0xa001). Vypočtená hodnota CRC je odeslána na konci bloku, nejdříve je odesláno osm vyšších bitů a poté osm nižších bitů hodnoty CRC.

3.1.10 Zpracování přijatých dat

Zpracování dat souvisí s globálním polem *buffer* pro příjem dat. Jde o 16 bajtové polem do kterého se ukládají přijaté data ze sériového portu. Pro příjem dat je použito přerušení od sériového portu, protože je třeba rychle reagovat na přijatá data, jinak by mohlo dojít ke ztrátě dat. Když, je tedy dokončen příjem jednoho bajtu nastává přerušení a program skočí do metody pro obsluhu přerušení: *ISR(USART_RXC_vect)*. V obsluze pro přerušení se kontroluje, zda jsou zpracované přijaté data (*noveData*). V případě že jsou zpracované, bajt ze sériového kanálu se zapíše do pole *buffer* na pozici

iBuffer a dále se kontroluje, jestli je blok kompletní porovnáním hodnoty *iBuffer* a délky přijímaných dat. Jestli jsou data kompletní, nastaví se hodnota proměnné *noveData* na 1 a nové data se nebudou přijímat, dokud metoda *zpracujData* data nezpracuje a nenastaví proměnno *noveData* na hodnotu 0.

```
ISR(USART_RXC_vect)
{
	if(noveData==0)
	{
		buffer[iBuffer]=UDR;
		if(iBuffer>2)
		{
			char delka=buffer[2];
			if(iBuffer==4+delka) noveData=1;
		}
		iBuffer++;
		if(iBuffer>15) iBuffer=15;
	}
}
```

Data, které teploměr přijímá, se týkají jeho nastavení a to buď nastavení toho co se bude zobrazovat na LCD displeji nebo práce se skupinami.

V případě nastavení zobrazení byl poslán řídicí příkaz: 0x0101 za ním následuje délka dat (6 bajtů) a za ní následuje 6 bajtů ve stejném formátu jako globální pole *cidlaLCD*. Po datech následuje CRC dat a v případě, že po výpočtu CRC dat a porovnáním s přijatým CRC se shodují, budou hodnoty pole *cdilaLCD* nahrazeny přijatými hodnotami.

Při definování nové skupiny v PC aplikaci a následném uložení se teploměru pošlou informace o nové skupině s řídicím příkazem: 0x0202. Na začátku se kontroluje, pomocí CRC zde data přišla v pořádku a v případě že ano přidá se v teploměru nová skupina. To začne dynamickým vytvořením skupiny datového typu *SKUPINA*:

```
SKUPINA *s = (SKUPINA*)malloc(sizeof(SKUPINA));
```

Po úspěšném vytvoření skupiny (funkce malloc nevrátila hodnotu NULL) je třeba definovat počet a pořadová čísla čidel. Tato informace je obsažena v datech poslaných PC aplikací. Data jsou složena z jednoho bajtu udávající počet čidel a za ním následují bajty, v nichž jsou uvedena pořadová čísla čidel skupiny. Přiřadíme tedy skupině počet čidel a pokusíme se alokovat prostor pro pole obsahující pořadová čísla čidel:

```
s->pocet=buffer[3];
s->cidla=malloc(buffer[3]*sizeof(char));
```

V případě že se podařilo alokovat prostor (funkce malloc nevrátila hodnotu NULL) načteme do nově alokovaného pole hodnoty poslané PC aplikací:

```
if(s->cidla!=NULL)
{
	for(char i=0; i<buffer[3];i++) s->cidla[i]=buffer[4+i]; }
```

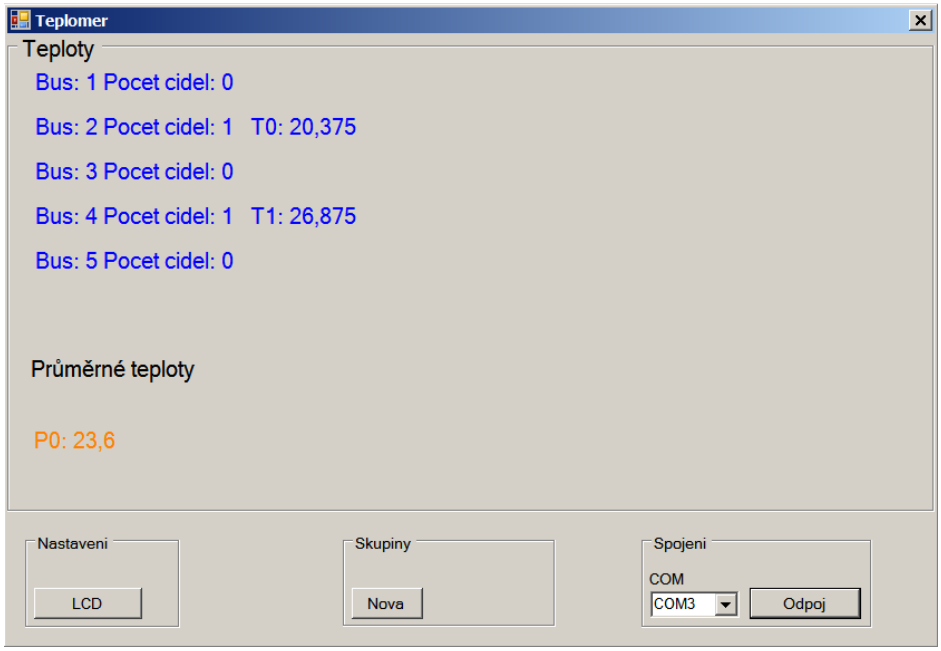
Nakonec je potřeba nastavit ukazatel poslední skupiny v seznamu na nově vytvořenou skupinu. Postupně tedy procházíme již vytvořené skupiny, dokud nedojdeme na konec a poslední skupině nastavíme ukazatel: *dalsi* na nově vytvořenou skupinu a zvýšíme celkový počet skupin (*pocSkupin*).

```
SKUPINA *s2=skupiny;
while(s2->dalsi!=NULL) { s2=s2->dalsi; }
s2->dalsi=s;
pocSkupin++;
```

4 PC aplikace

Počítačová aplikace je napsaná v programovacím jazyce C#. Aplikace poskytuje zobrazování změřené teploty každého čidla a také průměrné teploty definovaných skupin. Dále slouží k nastavení teploměru. Nastavením můžeme definovat, co chceme zobrazovat na LCD displeji teploměru. Můžeme zobrazit čtyři teploty, mohou to být teploty z vybraných čidel nebo průměrné teploty z definované skupiny čidel. Skupiny čidel, ze kterých se budou počítat průměrné teploty, se také definují v této aplikací.

Komunikace PC aplikace s teploměrem je realizována přes virtuální sériový port, který je vytvořen převodníkem FT232RL. Aplikace zachytává data na sériovém portu. Po přijetí bloku dat zkontroluje data pomocí CRC. V případě, že jsou přijatá data v pořádku, dále s nimi pracuje. Aplikace zpracovává všechny zasílané zprávy teploměrem. Když jde o zprávu, ve které je uložena informace o teplotě, aplikace si teplotu uloží a následně zobrazí. Na obrázku 4-1 je okno aplikace, kde jsou zobrazeny teploty dvou čidel a jedna průměrná teplota ze skupiny čidel.



Obrázek 4-1: PC aplikace

5 Závěr

Tato bakalářská práce byla pro mě hodně zajímavá, protože jsem si vyzkoušel hned několik různých prací. Postupně jsem prošel od návrhu elektronického zapojení teploměru, návrhu plošného spoje, konstrukce zařízení až k programování. Zapojení jsem vytvářel na počítači, v programu Eagle. Po vytvoření schématu jsem pomocí tohoto programu začal navrhovat desku plošného spoje podle schématu. Když byl návrh hotový, tak jsem foto cestou udělal DPS a následně zapájel všechny součástky. Potom bylo třeba napsat program pro mikrokontroler. Vývoj programu mi trval asi nejdéle. Program jsem psal postupně, vždy po malých částech, které bylo vždy třeba vyzkoušet, případně opravit a odladit. Program pro mikrokontroler byl psaný v jazyce C, který je celkem jednoduchý, ale v případě používání ukazatelů se dají lehce udělat chyby. Když jsem měl program téměř hotový, začal jsem pracovat na PC aplikaci. PC aplikace zachytává data na sériovém portu a následně zpracovává, případně zasílá data s nastavením teploměru. Myslím si, že komunikace mezi PC a teploměrem by se dala ještě rozšířit a také by se mohla vylepšit PC aplikace. Teploměr se mi podařilo dostat na konci do fáze, že po zapnutí napájení nebo při resetu si automaticky prohledá sběrnice a zapamatuje si unikátní kódy čidel, se kterými následně pracuje. Tady by se dal program ještě vylepšit, že by i v pozdějších fázích prohledával sběrnice a v případě nově nalezeného čidla by si jej taky zapamatoval a pracoval s ním. Dle počtu najitých čidel dále vytvoří defaultně jednu průměrovou skupinu z prvních dvou čidel, v případě že bylo připojeno více čidel. Následně čidla prochází, zjišťuje naměřené teploty, které si ukládá a ty podle nastavení zobrazuje na LCD displeji a odesílá přes UART. Přes PC aplikaci se mohou definovat nové průměrové skupiny, bohužel zatím s nimi program mikrokontroleru pracuje jenom v SRAM paměti a v případě resetu nebo po odpojení napájení jsou definované skupiny ztraceny, tento nedostatek by se dal vyřešit uložením všech konfiguračních hodnot do paměti EEPROM a vždy po resetu nebo po zapnutí napájení je načítat.

Seznam použité literatury

[1] Maxim Integrated Products [online]. USA : Maxim Integrated Products, 2008 [cit. 2010-11-07]. DS18B20 Programmable Resolution 1-Wire Digital Thermometer. Dostupné z WWW: <http://pdfserv.maximic.com/en/ds/DS18B20.pdf>.

[2] GM electronic [online]. ATM1602B. Dostupné z WWW: <http://www.gme.cz/_dokumentace/dokumenty/513/513-109/dsh.513-109.1.pdf>.

[3] Future Technology Devices International [online], 2011 [cit. 2010-06]. FT232R USB UART IC. Dostupné z WWW: <http://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT232R.pdf>.

[4] Atmel Corporation [online], 2011 [cit. 2011-02]. ATmega8L. Dostupné z WWW: <http://www.atmel.com/dyn/resources/prod_documents/doc2486.pdf>.

[5] GM electronic [online]. STMicroelectronics 2004 [cit. 2004-11]. L7800 SERIES POSITIVE VOLTAGE REGULATORS. Dostupné z WWW: <http://www.gme.cz/_dokumentace/dokumenty/330/330-149/dsh.330-149.1.pdf>.

[6] KADLEC, Václav. Učíme se programovat v jazyce C. Brno: CP Books, 2005. 277 s. ISBN 80-7226-715-9.

[7] digchip.com [online]. Samsung Semiconductor, Inc.[cit. 2000-06]. 40 Seg / 16 Com Driver & Controller For Dot Matrix LCD. Dostupné z WWW: <http://www.datasheetcatalog.org/datasheet/SamsungElectronic/mXruzuq.pdf>.

[8] Maxim Integrated Products [online].Book of iButton Standards [cit. 2002-01-16] Dostupné z WWW: <http://pdfserv.maxim-ic.com/en/an/AN937.pdf>